JÜLICH
FORSCHUNGSZENTRUM

Supercomputer architectures

An Introduction to HPC

3 Nov 2015 | Alexander Peyser | Simulation Lab Neuroscience
Jülich Supercomputing Centre
Institute for Advanced Simulation
Forschungszentrum Jülich

What is High Performance Computing?

Hardware architectures

Current Systems

Working with Supercomputers

Programming Paradigms

What is High Performance Computing?

- HPC definition
- HPC capabilities
- What do we have?
- How do we use them?
- Fully scaled projects

HPC definition

What is High Performance Computing?

JÜLICH FORSCHUNGSZENTRUM

What is a Supercomputer?

Wikipedia: A supercomputer is a computer with a high-level computational capacity compared to a general-purpose computer. Performance of a supercomputer is measured in floating point operations per second (FLOPS) instead of million instructions per second (MIPS). As of 2015, there are supercomputers which can perform up to quadrillions of FLOPS

HPC definition

What is High Performance Computing?

JÜLICH FORSCHUNGSZENTRUM

What is it in short?

A computer at the very limit of what is currently available

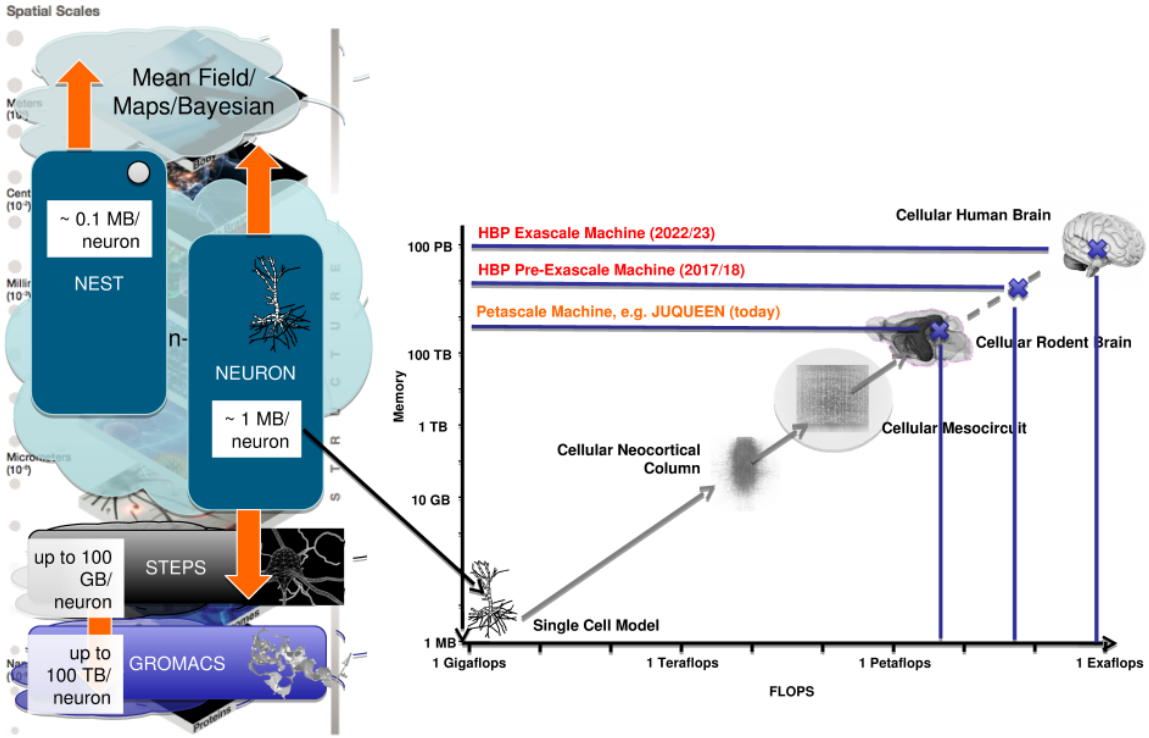
HPC definition

What is High Performance Computing?

JÜLICH FORSCHUNGSZENTRUM

What do we do with Supercomputers?

Supercomputers are massively parallel machines using the most advanced nodes, interconnects and memory to tackle problems that are too large or take too long for commodity computers



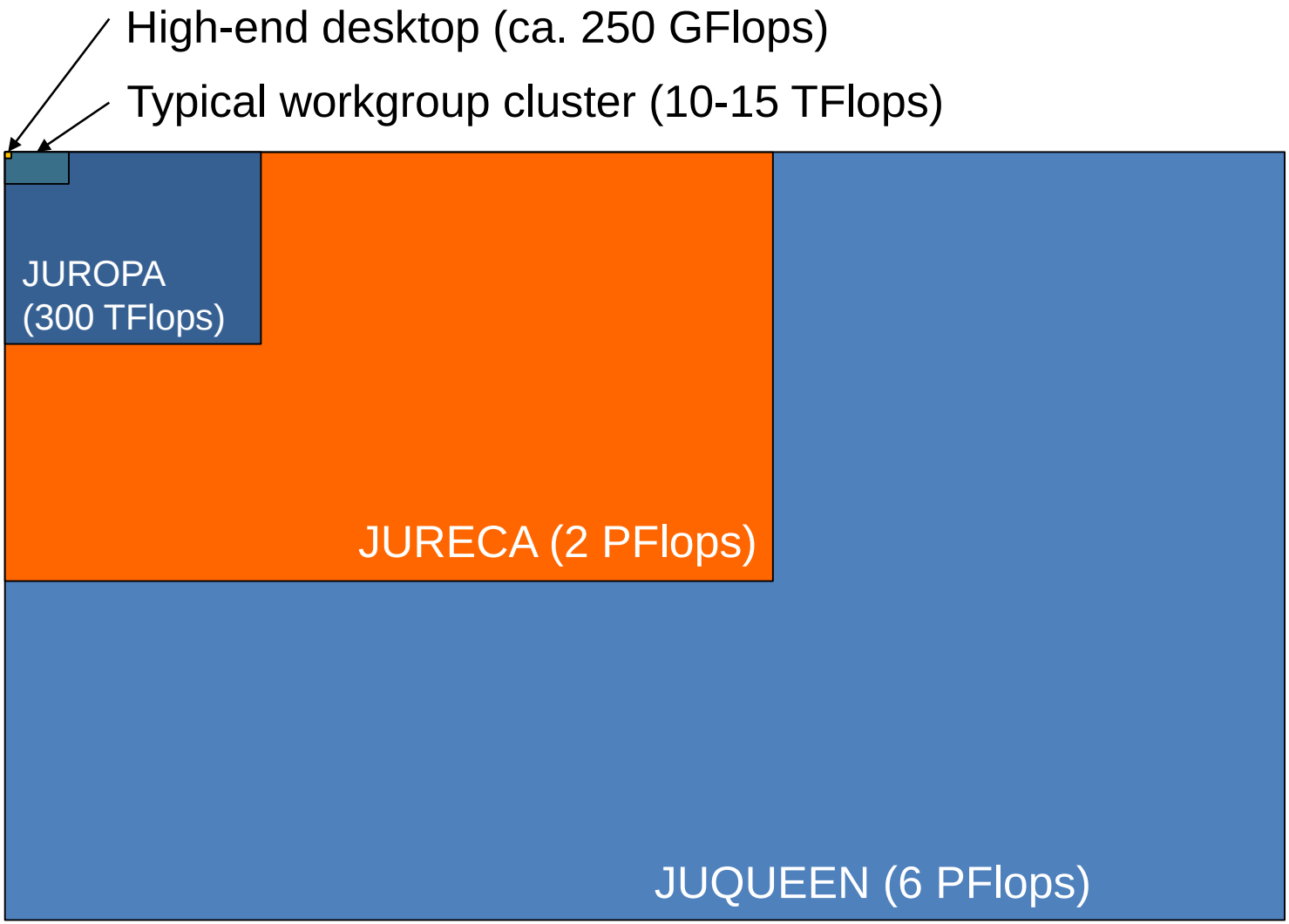
Felix Schürmann (EPFL)

HPC capabilities

What is High Performance Computing?

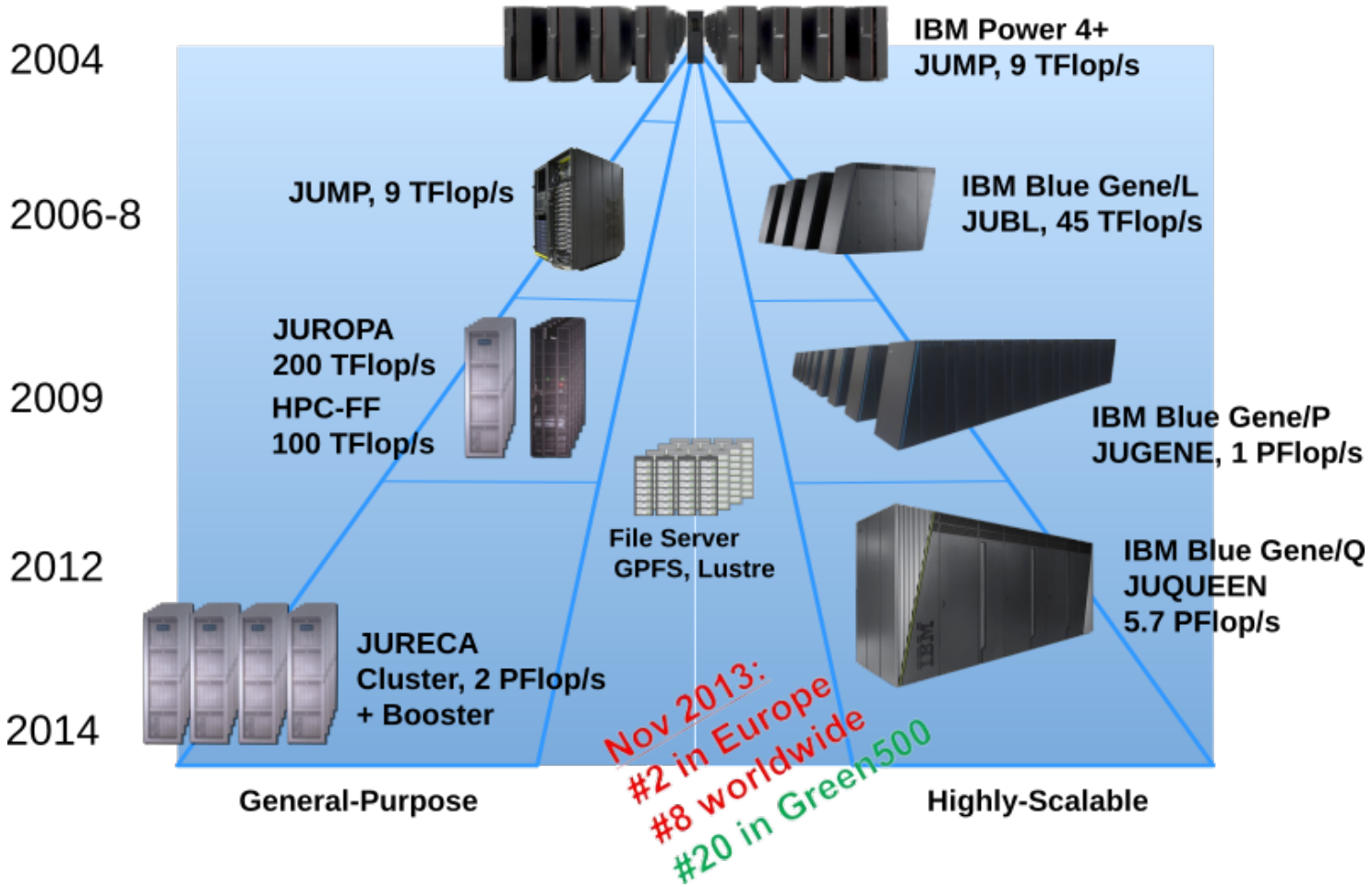
JÜLICH FORSCHUNGSZENTRUM

Theoretical peak performance



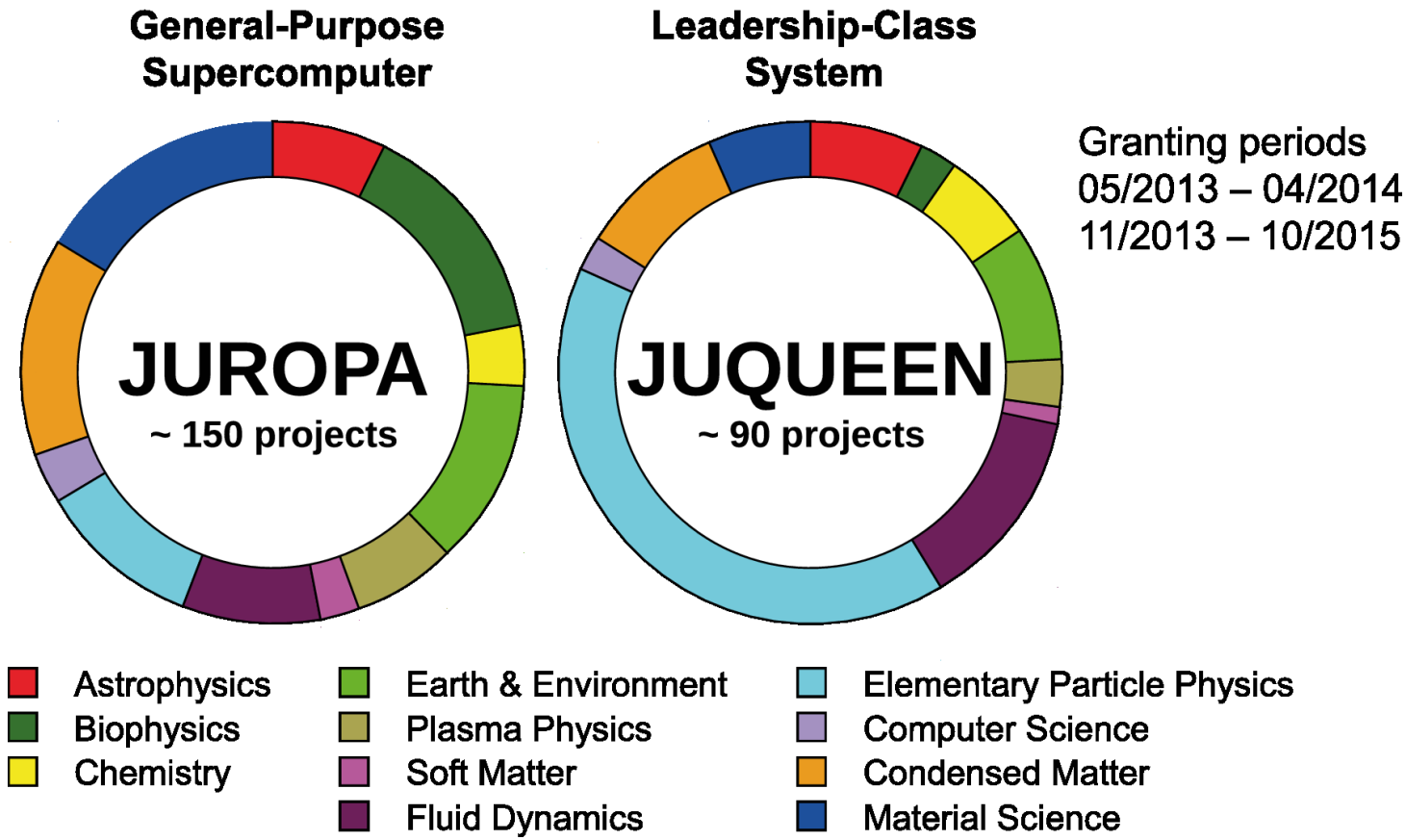
What do we have?

What is High Performance Computing?



How do we use them?

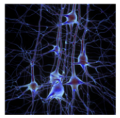
What is High Performance Computing?



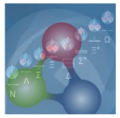
Fully scaled projects

What is High Performance Computing?

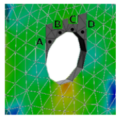
CIAO
Multiphysics, multiscale Navier-Stokes solver for turbulent reacting flows in complex geometries.



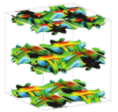
CoreNeuron
Simulating electrical activity of neuronal networks with morphologically-detailed neurons.



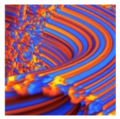
dynQCD
Lattice Quantum Chromodynamics (QCD) with dynamical fermions.



FE2TI (ex_nl/FE^2)
A scale-bridging approach incorporating micro-mechanics in macroscopic simulations of multi-phase steels.

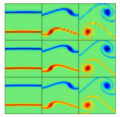


FEMPAR
A framework for the massively parallel FE simulation of multiphysics problems governed by PDEs.

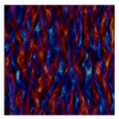


Gysela
A gyrokinetic code from CEA, France, for modelling fusion core plasmas.

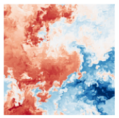
ICON
A solver for fully compressible non-hydrostatic equations of motion at very high horizontal resolution.



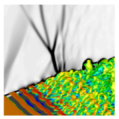
PMG+PFASST
A space-time parallel solver for systems of ODEs



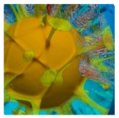
PP-Code
A particle-mesh code for simulating charged and neutral particle dynamics in relativistic and non-relativistic plasm



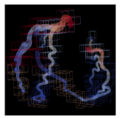
psOpen
Direct numerical simulation of fine-scale turbulence.



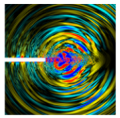
SHOCK
Direct numerical simulation of compressible flow.



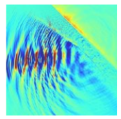
Terra-Neo
A multigrid solver for geophysics applications from the University of Erlangen.



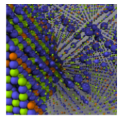
waLBerla
A widely applicable Lattice Boltzmann solver from the University of Erlangen.



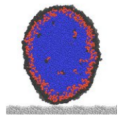
ZFS
A multiphysics framework for compressible and incompressible flow, aeroacoustics, and combustion phenomena.



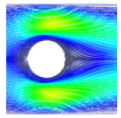
JuSPIC
A fully relativistic Particle-in-Cell code for laser-plasma simulations.



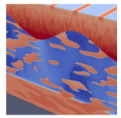
KKRnano
Korringa-Kohn-Rostoker Green function code for quantum description of nano-materials.



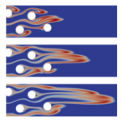
LAMMPS(DCM)
A Dynamic Cutoff Method for a classical molecular dynamics code, the Large-scale Atomic/Molecular Massively Parallel Simulator.



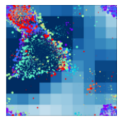
MP2C
Fluid simulations using a hybrid representation of solvated particles.



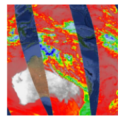
μφ (muPhi)
Water flow and solute transport in porous media.



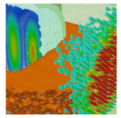
Musubi
A multicomponent Lattice Boltzmann solver for flow simulations.



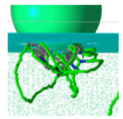
NEST
A simulator for spiking neural network models that focus on the dynamics, size and structure of neural systems.



JURASSIC
A fast solver for infrared radiative transfer in the Earth's atmosphere.



PEPC
A particle tree code for solving the N-body problem for Coulomb, gravitational and hydrodynamic systems.



IMD
A software package for classical molecular dynamics simulations.

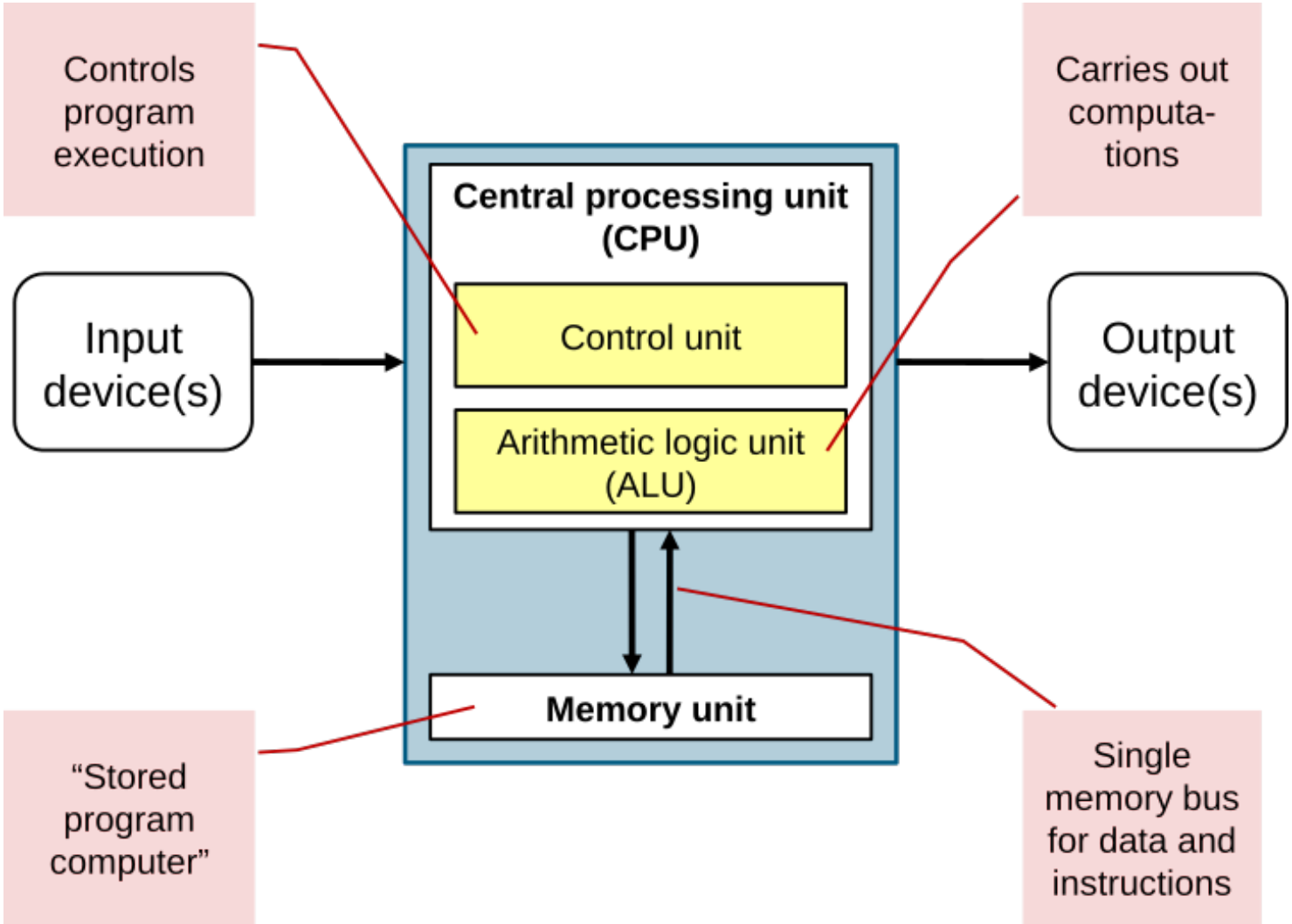
OpenTBL
Direct numerical simulation of turbulent flows.

Hardware architectures

- Von Neumann Architecture
- Von Neumann Bottlenecks
- Parallelizing out of the bottlenecks
- Parallelism speedup
- General implemenation
- GPUs: data parallel problems

Von Neumann Architecture

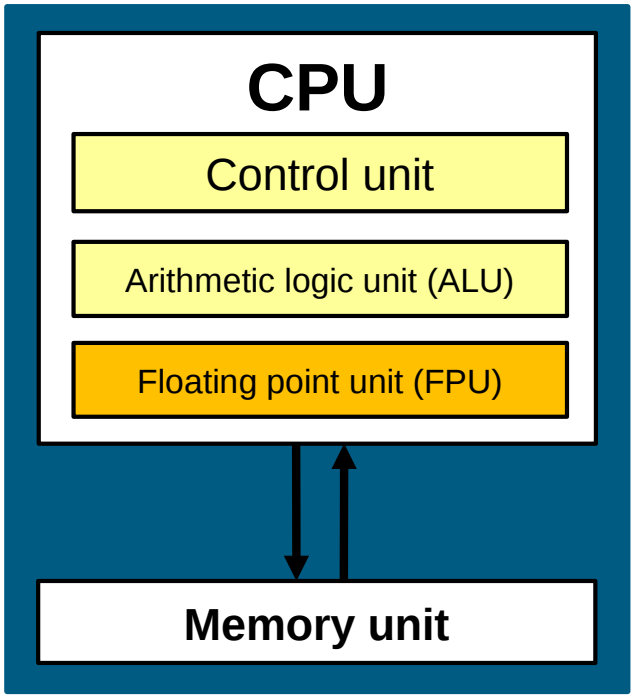
Hardware architectures



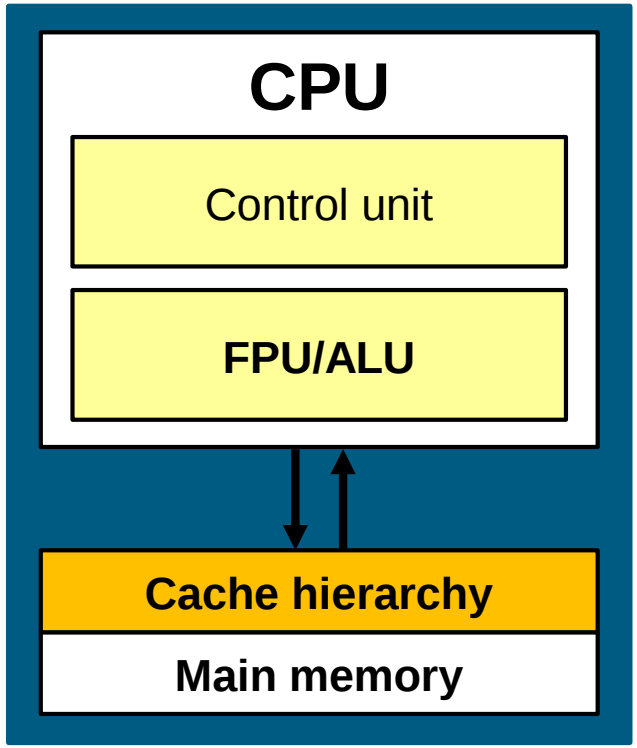
Von Neumann Bottlenecks

Hardware architectures

Computing with real numbers



Enhancing memory access



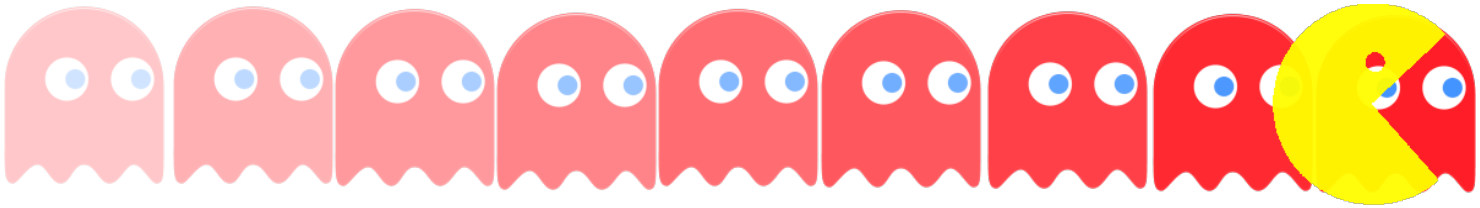
Flops	floating point operations per second
Bandwidth	Number of bytes of memory transferred per second
Latency	Time delay to completion of a single computational or memory access operation

Parallelizing out of the bottlenecks

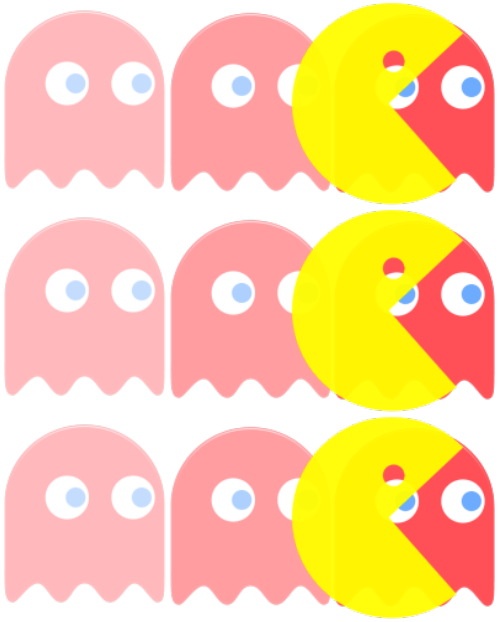
Hardware architectures

JÜLICH FORSCHUNGSZENTRUM

One pacman eats nine ghosts in 3 seconds...



... but three pacmen eat 9 ghosts in 1 second



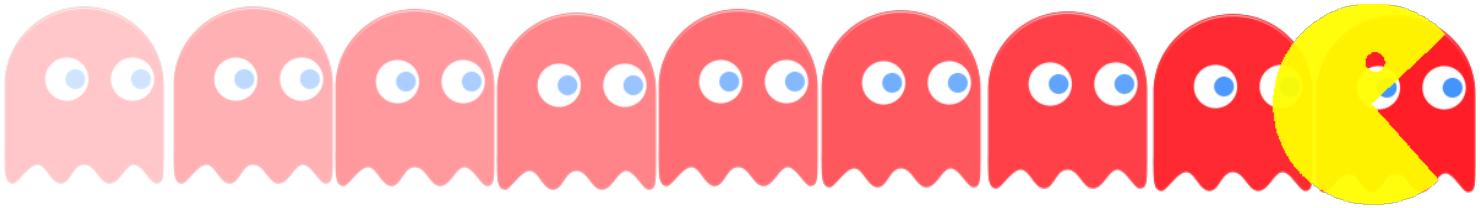
... which is strong scaling

Parallelizing out of the bottlenecks

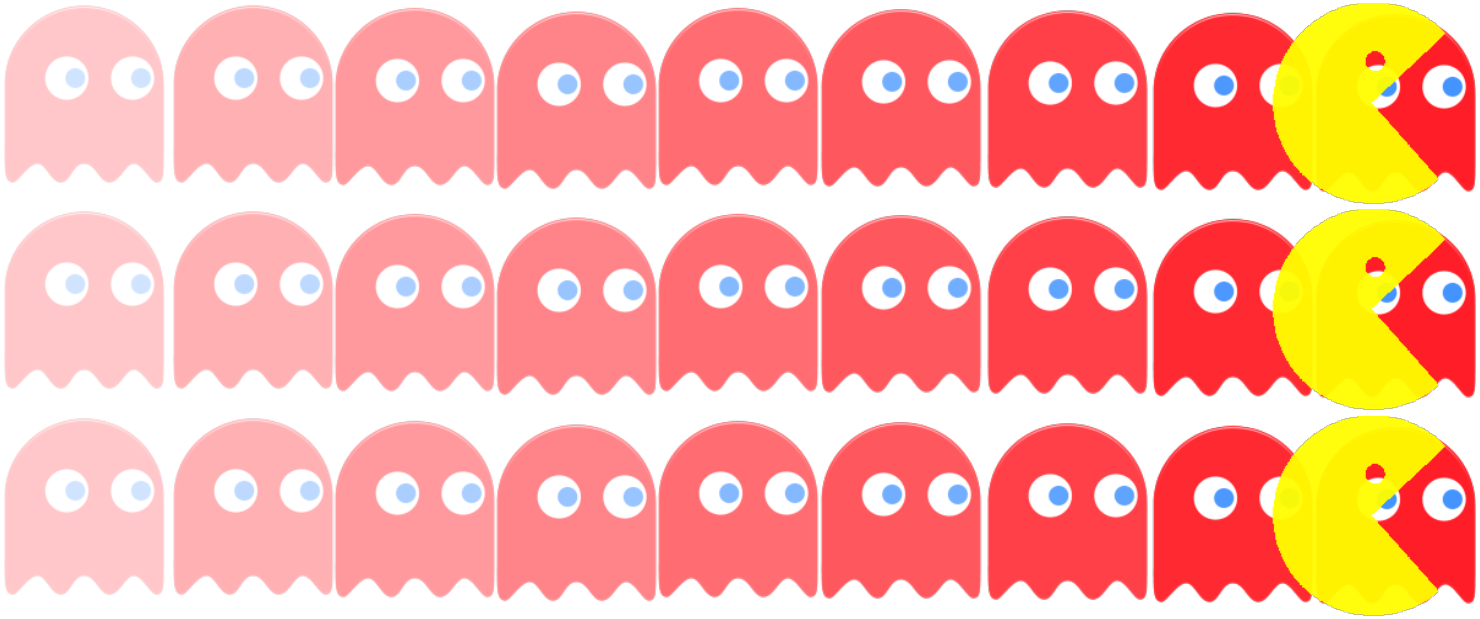
Hardware architectures

JÜLICH FORSCHUNGSZENTRUM

One pacman eats nine ghosts in 3 seconds...



... or three pacmen eat 27 ghosts in 3 seconds



... which is weak scaling

JÜLICH FORSCHUNGSZENTRUM

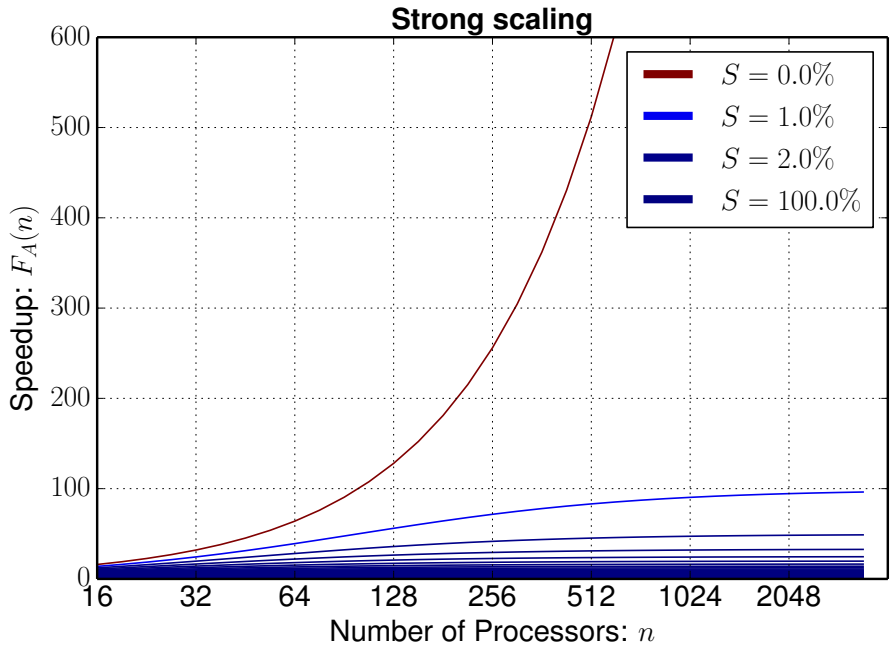
Limits to parallelism

Amdahl's law: $F_A(n) = \left[S + (1-S)/n\right]^{-1}$

If $S > 0$, $\lim_{n\to\infty} F_A(n) = 1/S$

Gustafson's law: $F_G(n) = S + n * (1-S)$

$(F_G(n) - S)/n = 1 - S$



Parallelism speedup

Hardware architectures

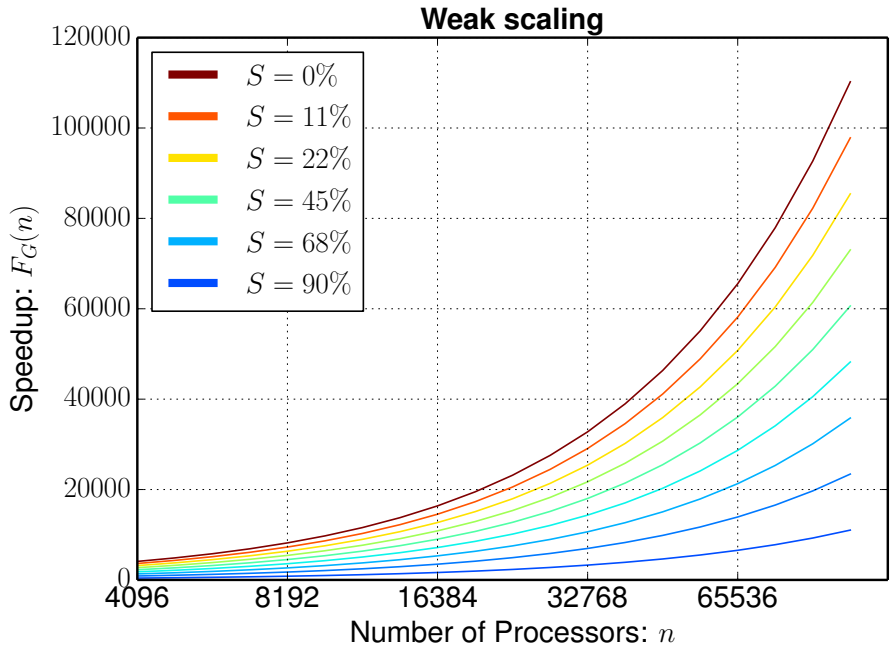
Limits to parallelism

Amdahl's law: $F_A(n) = \left[S + (1-S)/n\right]^{-1}$

If $S > 0$, $\lim_{n\to\infty} F_A(n) = 1/S$

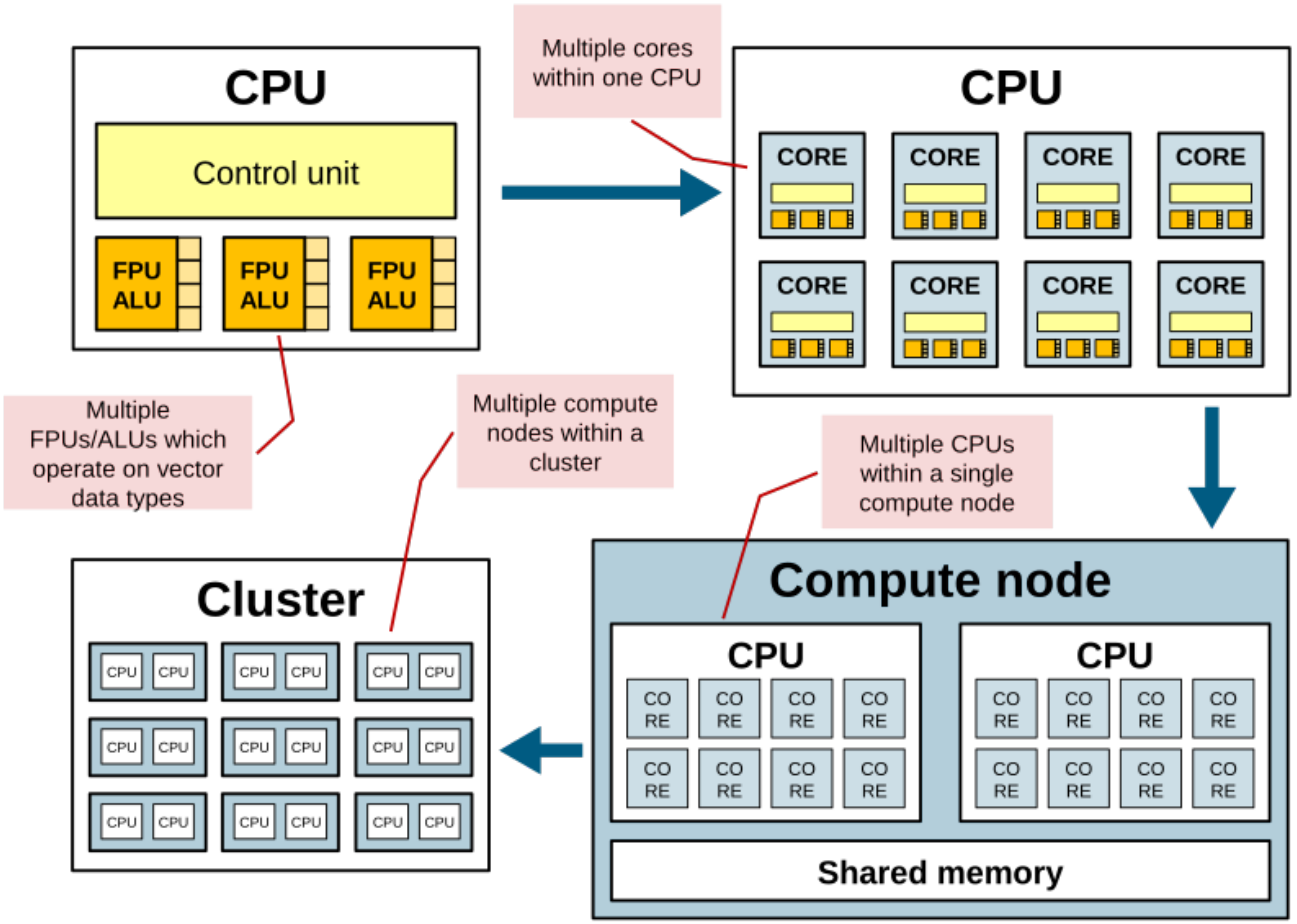
Gustafson's law: $F_G(n) = S + n * (1-S)$

$(F_G(n) - S)/n = 1 - S$



General implemenation

Hardware architectures

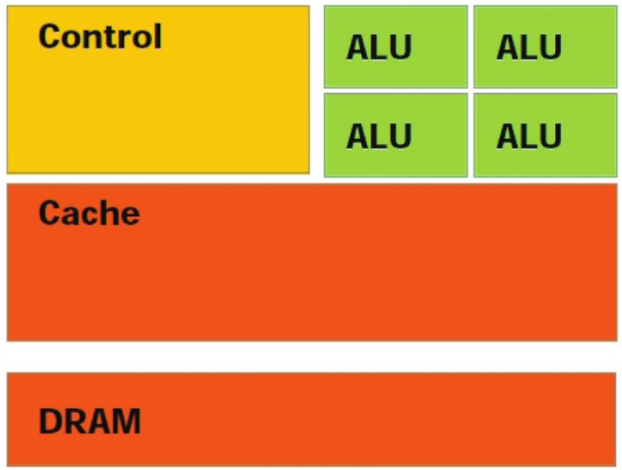


GPUs: data parallel problems

Hardware architectures

GPU: Graphics Processing Unit

A large number of arithmetic/floating point units with reduced control logic in parallel to the CPU. Originally developed for 3D graphics rendering



CPU

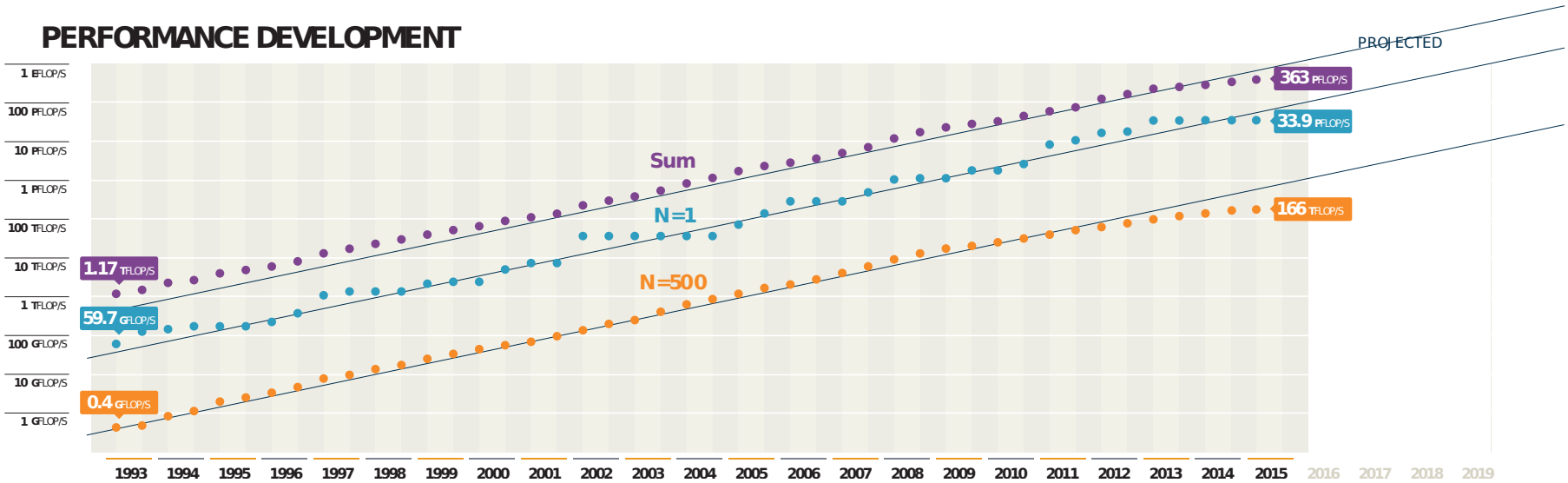
GPU

nvidia, amd

Current Systems

- Performance
- Architecture
- Accelerators
- JUQUEEN
- JURECA

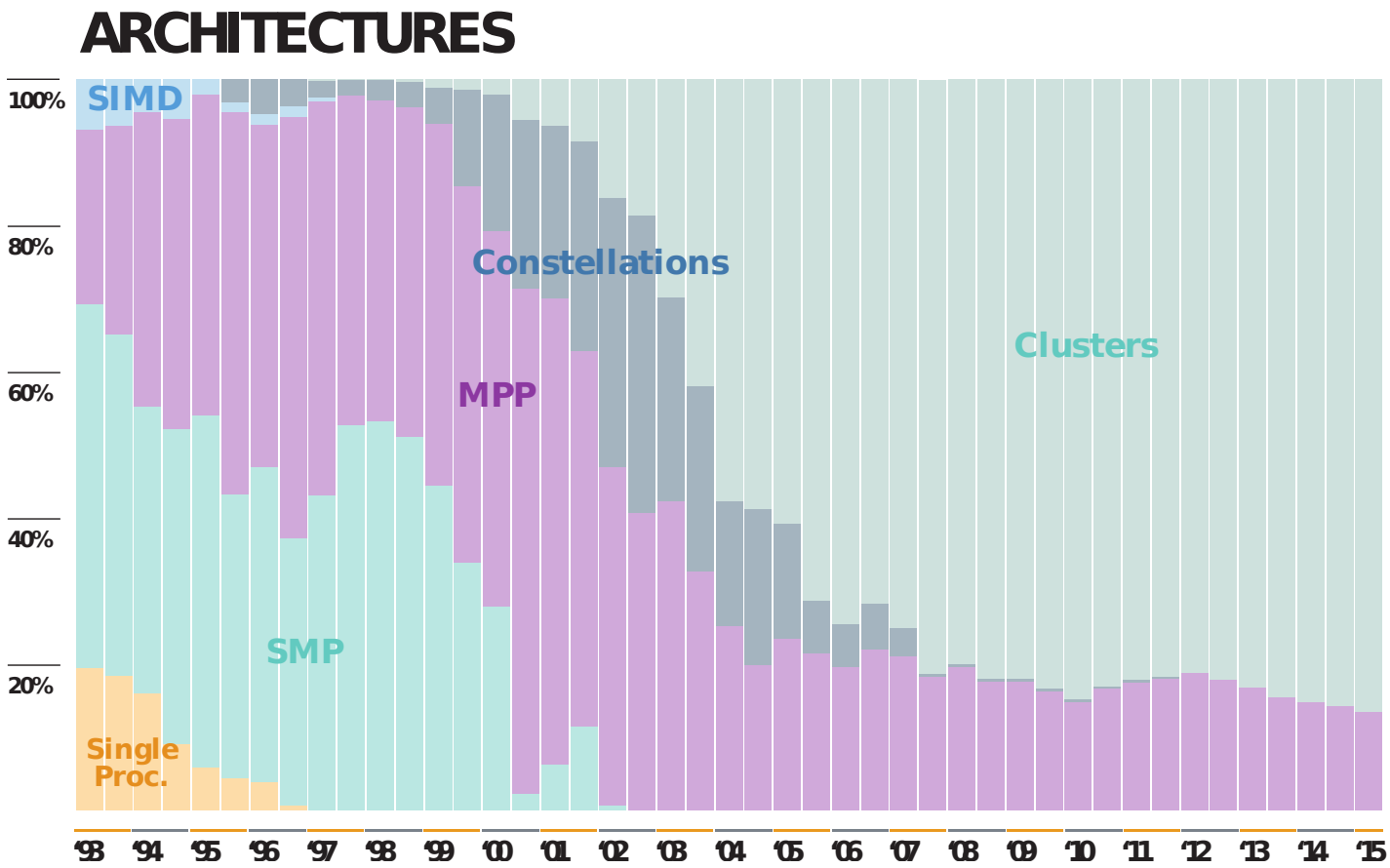
Performance
Current Systems



top500.org

Architecture

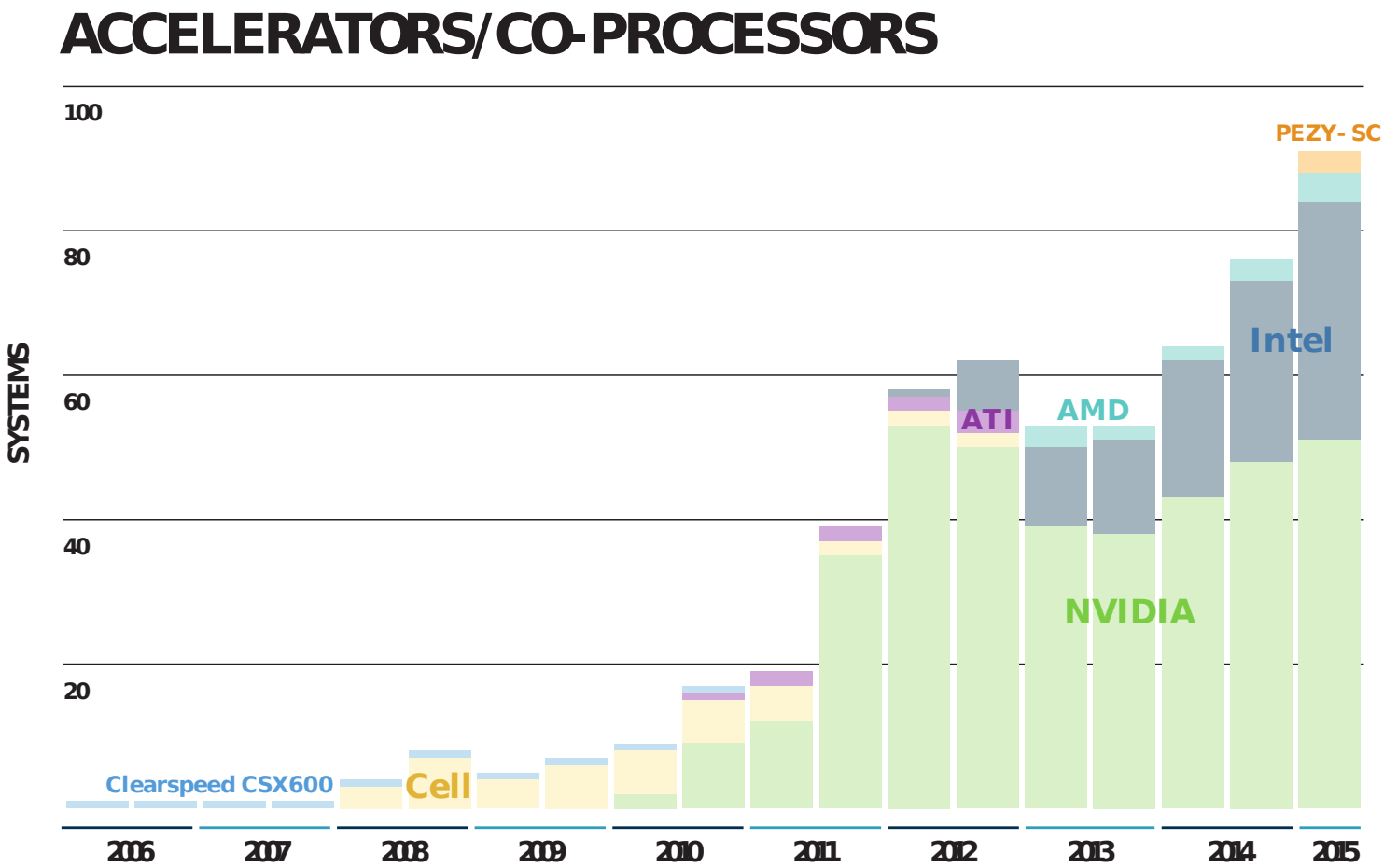
Current Systems



top500.org

Accelerators

Current Systems



top500.org

JUQUEEN

Current Systems

JÜLICH FORSCHUNGSZENTRUM

Data Sheet	
Manufacturer	IBM
Design	Bluegene/Q
CPU Core	PowerPC A2 16 (x4)
Cores	458,752
Linpack Perf. (R_{max})	5.009 PFlop/s
Theoretical Peak (R_{peak})	5.872 PFlop/s
Power	2.3 MW
Memory	458 TB
Interconnect	Custom Interconnect Network
Topology	5D Torus
Operating System	Linux / CNK
Batch	LoadLeveler

JURECA

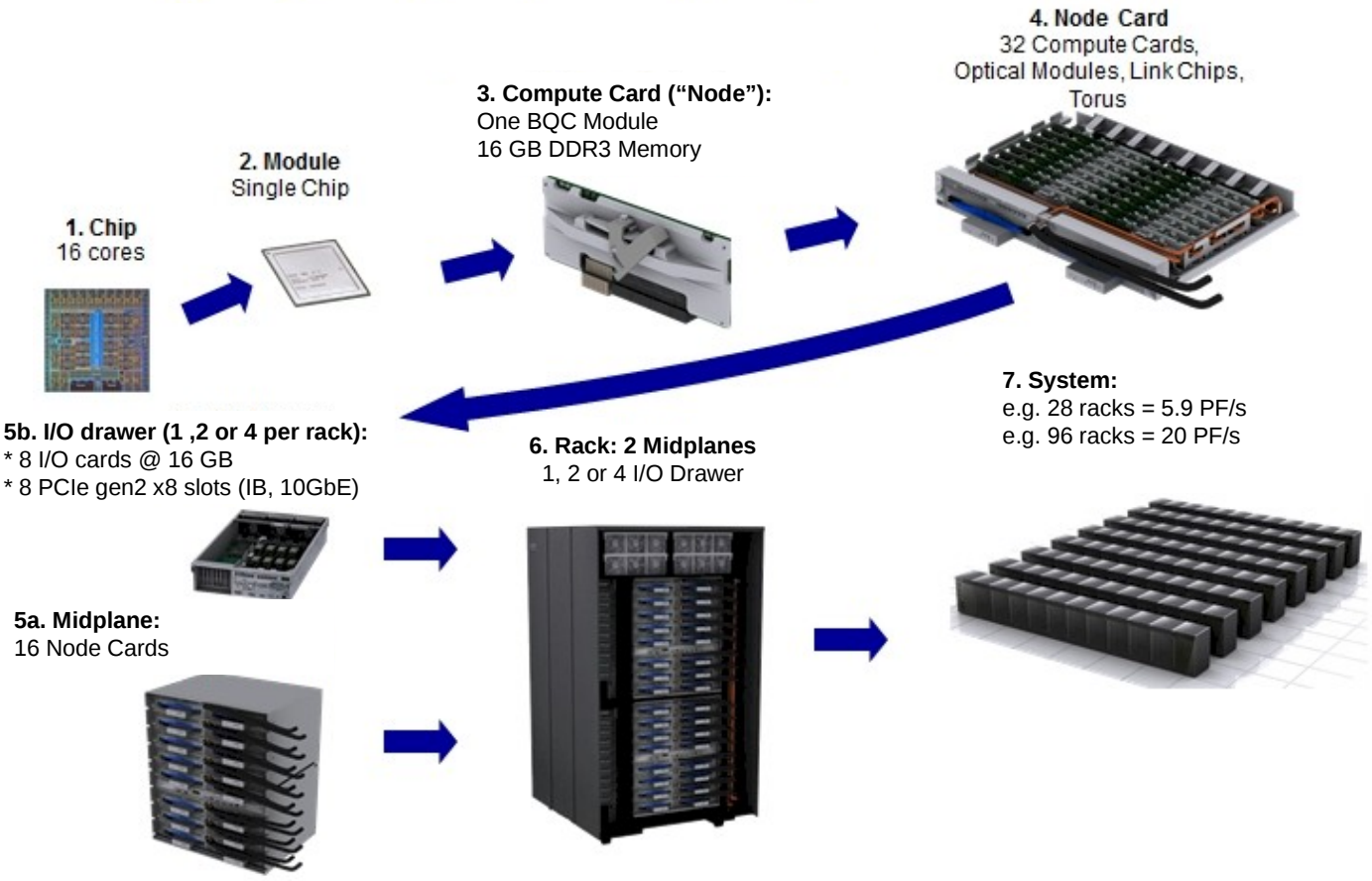
Current Systems

Data Sheet	
Manufacturer	T-Platforms, ParTec, Intel, Mellanox
Design	Intel / GPU
CPU Core	2 Intel Haswell 12-core & K80 GPUs
Cores	45,216
GPUs	75 nodes w/ 2 K80 (4992 CUDA)
Linpack Perf. (R_{max})	(Prel) 1.4 PFlop/s
Theoretical Peak (R_{peak})	1.8 PFlop/s + 0.44 (GPU)
Power	???
Memory	281 TB
Interconnect	Mellanox EDR InfiniBand
Operating System	CentOS 7 Linux
Batch	Slurm

Working with Supercomputers

- Hardware
- Cluster organization
- Running system
- LoadLeveler commands
- LoadLeveler script

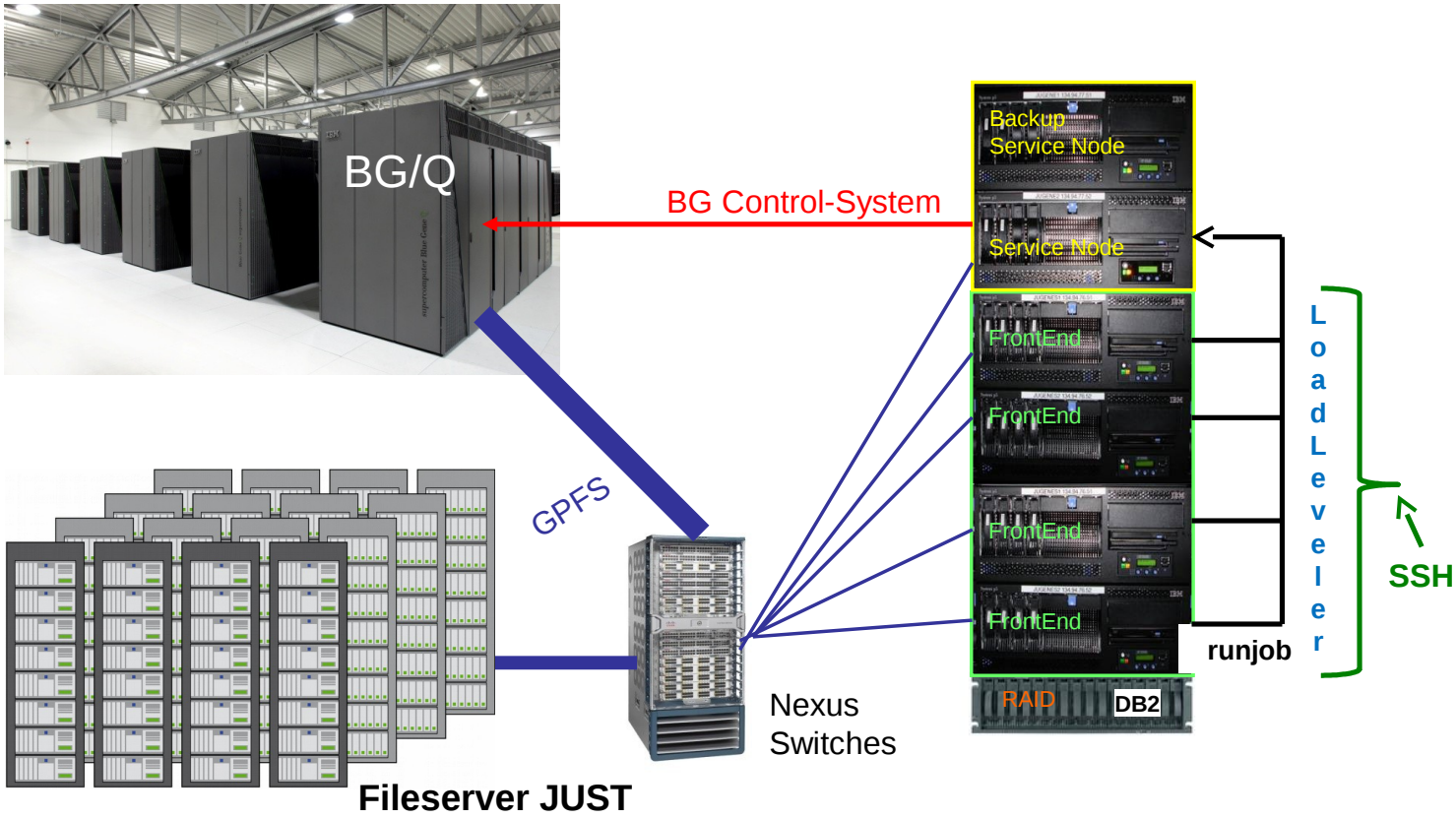
Hardware
Working with Supercomputers



IBM

Cluster organization

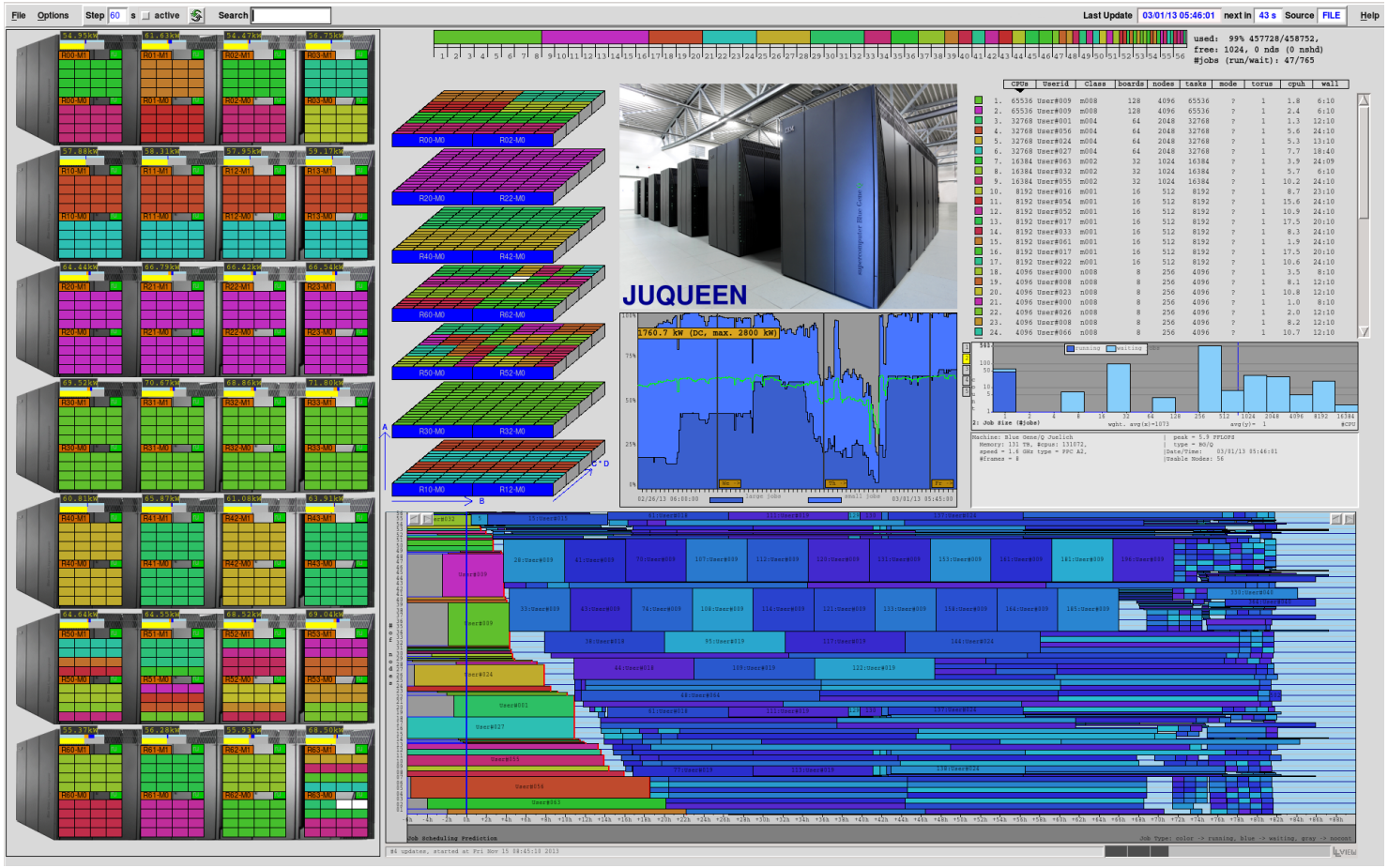
Working with Supercomputers



Michael Stephan & Jutta Doctor (JSC)

JÜLICH FORSCHUNGSZENTRUM

Supercomputers are shared resources



LoadLeveler commands

Working with Supercomputers

Command	Description
`llsubmit <jobfile>`	Send job to the queuing system
`llq`	List all queued and running jobs
`llq -l <job ID>`	detail information about the specific job
`llq -s <job ID>`	details information about a specific queued job, such as start time
`llq -u <user>`	list all jobs from one user
`llcancel <job ID>`	Kill the specified job
`llstatus`	Display the status of LoadLeveler
`llclass`	List existing classes and their properties
`llqx`	Show detailed information about all jobs

LoadLeveler script

Working with Supercomputers

```
# @job_name          = weakScale_nmr00032_nthr04_N16667_dryRun
# @job_type          = bluegene
# @bg_size           = 32
# @bg_connectivity   = TORUS
# @environment       = COPY_ALL
# @wall_clock_limit  = 00:30:00
# @output            = $(HOME)/log/juqueen/$(job_name).$(jobid).out
# @error             = $(HOME)/log/juqueen/$(job_name).$(jobid).err
# @notification      = error
# @notify_user       = w.schenck@fz-juelich.de
# @queue

export OMP_NUM_THREADS=4

NPROCS=128
NEST_VERSION=10kcollaps_dry_run
NEST_EXE=${HOME}/opt/NEST/juqueen_${NEST_VERSION}/bin/nest
SLI_SCRIPT=spike_counting_4G_dryRun.sli

runjob --exe $NEST_EXE \\
       --np $NPROCS \\
       --ranks-per-node 1 \\
       --verbose 1 --exp-env OMP_NUM_THREADS \\
       --args $SLI_SCRIPT
```

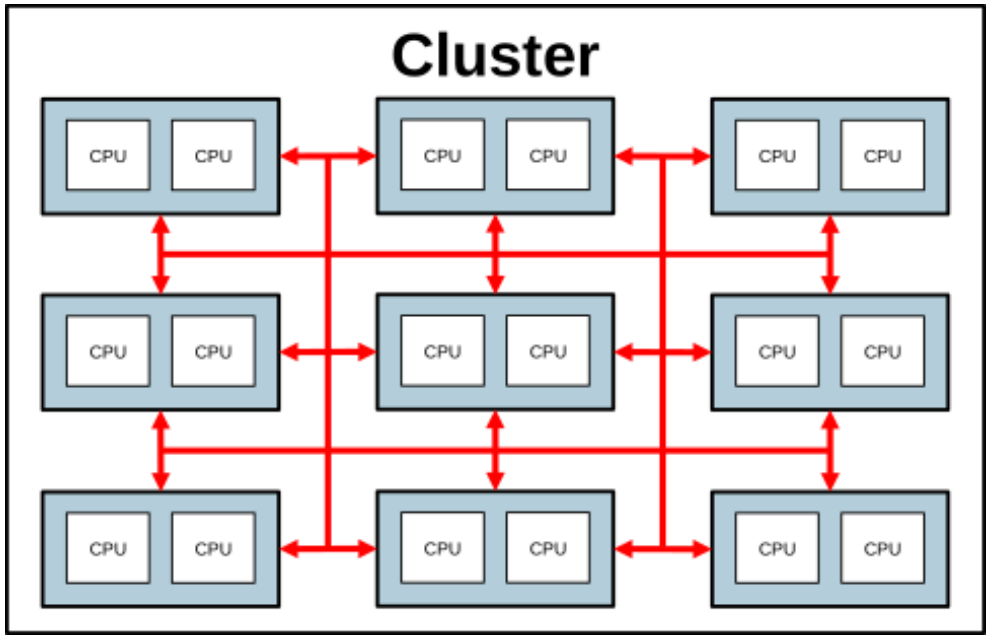
JÜLICH
FORSCHUNGSZENTRUM

Programming Paradigms

- Overview
- Languages
- Tools

Overview

Programming Paradigms



Decomposition

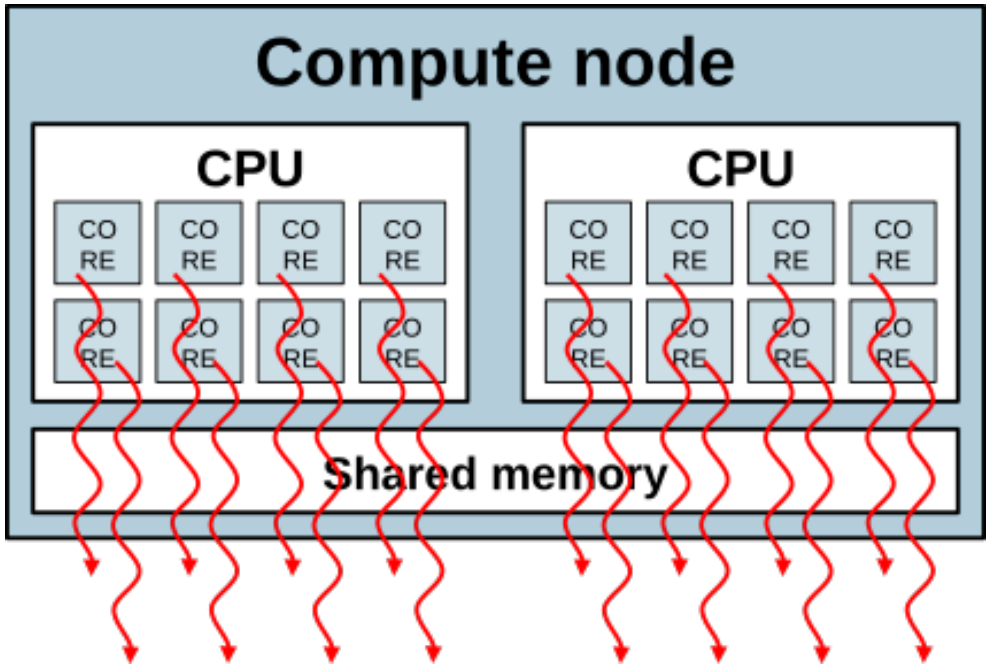
MPI At the cluster level, message passing standard via network between nodes (application)

OpenMP At the node level, thread programming standard using shared memory (process)

OpenCL At the GPU level, work groups of parallel kernels

Overview

Programming Paradigms



Decomposition

MPI At the cluster level, message passing standard via network between nodes (application)

OpenMP At the node level, thread programming standard using shared memory (process)

OpenCL At the GPU level, work groups of parallel kernels

Overview

Programming Paradigms

Decomposition

MPI At the cluster level, message passing standard via network between nodes (application)

OpenMP At the node level, thread programming standard using shared memory (process)

OpenCL At the GPU level, work groups of parallel kernels

JÜLICH FORSCHUNGSZENTRUM

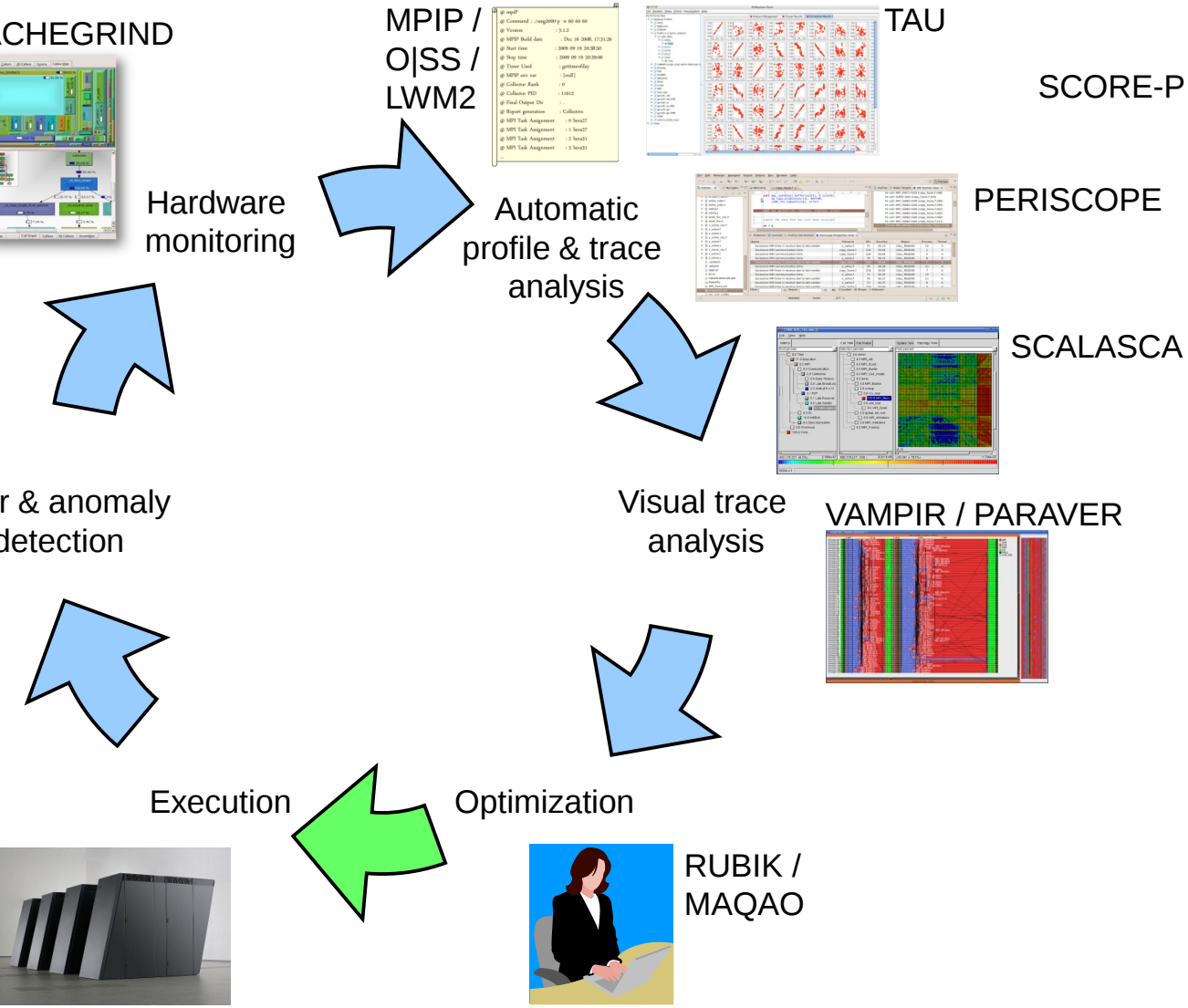
Which programming languages can I use?

Scripting	Run over parameter spaces with scripting languages and JUBE — and run (most) any code
MPI	C/C++/Fortran/Python/JAVA... (special library)
OpenMP	C/C++/Fortran (special directives)
pthreads	Native threads for most languages
GPU	C/C++ with other experimental kernel mappings, bindings on cpus to most everything... (kernels compiled and shipped)

Make the outer loops & code as you want, and focus on optimizing core kernels

Tools

Programming Paradigms



JÜLICH
FORSCHUNGSZENTRUM

Thank You!